An Algorithm for Parallelizing Sequential Minimal Optimization

Xinyue Wang and Jun Guo

Computer Center
East China Normal University
3663 Zhong Shan Rd. N., Shanghai, China
jguo@cc.ecnu.edu.cn

Abstract. In this paper, a new algorithm for training support vector machines (SVMs) for classification problems with parallel sequential minimal optimization (SMO) is proposed. The selection of the working set is paralleled so that the iteration of the optimization process is reduced greatly. The experimental results show the training time of the proposed method is always less than the original SMO algorithm, and at the same time the classification accuracy is kept.

Keywords: SVM, SMO, Parallelization.

1 Introduction

Support Vector Machines (SVM) is a very useful tool for solving pattern recognition problems [1]. Due to its good performance in generalization on the basis of statistical learning theory [2], more and more research on SVM has been done and a lot of variants of SVM have been implemented.

The training step of SVM in the dual space was proposed in [1] using Lagrange multipliers. With the increasing number of the training data and support vectors, more optimization strategies have been adopted, especially some decomposition methods such as chunking[1], Osuna's algorithm, and Sequential Minimal Optimization(SMO) [3, 4].

The chunking method starts with a subset of data called chunks and iteratively enlarges that subset by including those examples that violate the optimization conditions. In Osuna's algorithm, the training examples were separated into two groups: the working set denoted as B and the rest of the data defined as N. In [4], Osuna's theorem demonstrates that moving a variable from set B to set N does not change the cost function and ensures that there is a strict improvement in the cost function after moving a variable that violates the optimality condition. Osuna's algorithm makes training SVMs tractable especially when the number of support vectors is quite large. SMO [3] ensures its convergence due to Osuna's theorem and optimizes only two Lagrange multipliers at a time, which means that the size of working set is only two. The performance of SMO in training time is better than Osuna's algorithm according to most of the experiments in [3]. This leads to a better performance in many circumstances and can be applied to both classification and regression problems. There are also some other algorithms using decomposition methods like $\text{SVM}^{\text{Light}}$[5] and the algorithm used in LIBSVM [6, 7]. The former utilizes a first-order approximation of the object function and seeks a steepest direction of descent. The latter considers a more precise approximation using second order information, aiming at obtaining a faster convergence speed [6].

There are some researches about the parallelizing SMO algorithm, reducing the training time of SVM considerably. In [8], the whole data set is divided into smaller subsets and distributed to different processors. By parallelizing the procedure of updating the error array denoted by F_i, the parallel SMO algorithm is much faster than the original one, especially when the number of data is large. The Casade SVM was introduced in [9], which trains SVMs in layers like filters. Because some examples are unlikely to be support vectors in an early stage of optimization, they are filtered. Those being support vectors are passed down to another layer for further training and testing. The process iterates until convergence. The main idea in parallel SVM is to reduce the time and memory cost by training, which derives our parallel algorithm dealing with working set selection.

In this paper, an algorithm of parallelizing the procedure in training SVM is proposed. We focus on the selection of working sets and choose two pairs at the same time. From the experiment, we show that the time of training is less than the original serial SMO. Furthermore, the prediction accuracy is maintained.

This paper is organized as follows: section 2 gives a brief introduction to SVM, decomposition methods and some related works. In section 3, we propose our parallelizing solutions and the experiment results are shown in section 4. Finally, section 5 discusses some problems in our method and gives a possible solution in future works.

2 Support Vector Machines

In classification problems, we denote $\{X_i, y_i\}_{i=1}^{l}$ as input samples where $\{X_i\}_{i=1}^{l}$ are the training examples and $y_i = \{-1,1\}_{i=1}^{l}$ are the corresponding labels. Training an SVM aims at solving an optimization problem given below:

$$\min_{w} \quad f(w) = \frac{1}{2} w^T w \tag{1}$$

$$\text{subject to} \quad y_i(w^T x_i + b) \geq 1 \quad i = 1, 2, \ldots, l$$

In order to handle misclassifications when the input samples contain noises, slack variables ξ_i are added into (1). Thus we obtain a soft margin classification problem.

$$\min_{w} \quad f(w) = \frac{1}{2} w^T w + C\sum_{i=1}^{l} \xi_i \tag{2}$$

$$\text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, \ i = 1, 2, \ldots, l$$

where C > 0 is the regularization parameter. One way of solving (2) is to handle with the following dual problem based on Lagrange duality theorem and get the following optimization problem in the dual.

$$\max_{\alpha} \quad Q(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \varphi^T(x_i)\varphi(x_j) \tag{3}$$

$$\text{subject to} \quad \sum_{i=1}^{l} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C \ \ i = 1, 2, \ldots, l$$

where α_i is the Lagrange multiplier and $\varphi(x)$ is the mapping function and we denote $K(x, x') = \varphi^T(x)\varphi(x')$ known as the kernel function which implicitly maps data to a high-dimensional space without computing the mapping function. There are some typical kernel functions such as polynomial kernel and Gaussian kernel. Finally, if the kernel matrix satisfies Mercer's condition which means that K is positive semi-definite, (1) becomes a QP optimization problem which contains no local minima.

3 Sequential Minimal Optimization

However, the Kernel matrix $y_i y_j K(x_i, x_j)$ may be too costly to be fit into the memory due to its size and density. Platt's sequential minimal optimization (SMO) [3] focuses on choosing two instances as working set. However, due to its underlying inefficiency in choosing the threshold value, modifications of Platt's SMO are proposed in [10], which introduces an improvement by choosing and updating the worst violating pair and also deals with two threshold parameters. The optimization problem is defined as follows. First, we rewrite the optimization problem from (3).

$$\max_{\alpha} \quad Q(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{4}$$

$$\text{subject to } \sum_{i=1}^{l} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \quad i = 1, 2, \ldots, l$$

According to [10], we define:

$$I_{up}(\alpha) = I_0(\alpha) \cup I_1(\alpha) \cup I_2(\alpha)$$

$$I_{low}(\alpha) = I_0(\alpha) \cup I_3(\alpha) \cup I_4(\alpha)$$

where

$$\begin{aligned} I_0(\alpha) &= \{i : 0 < \alpha < C\} \\ I_1(\alpha) &= \{i : y_i = 1, \alpha_i = 0\} \\ I_2(\alpha) &= \{i : y_i = -1, \alpha_i = C\} \\ I_3(\alpha) &= \{i : y_i = 1, \alpha_i = C\} \\ I_4(\alpha) &= \{i : y_i = -1, \alpha_i = 0\} \end{aligned}$$

The optimization condition for the problem is the following:

$$\min_{i \in I_{up}(\alpha)} F_i(\alpha) \ge \max_{j \in I_{low}(\alpha)} F_j(\alpha) \tag{5}$$

where $F_i(\alpha) = y_i \dfrac{\partial Q(\alpha)}{\partial \alpha_i} = y_i(\sum_{j=1}^{l} \alpha_j y_i y_j K(x_i, x_j) - 1)$. Thus, if a pair of indices (i, j) violates the above optimization condition, it is called a violating pair. Due to [10], the optimality holds if and only if no violating pair exists.

4 Parallelizing the Process of Working Set Selection

In this section, the parallel algorithm is proposed. First, we briefly introduce the idea of our algorithm and its difference from some other solutions. Then we give a more detailed picture of how to perform our parallelization on choosing violating pairs in working set selection.

Different from the methods above, we parallelize the process of working set selection and the updates of Lagrange multipliers in LIBSVM. In one of the parallelizing processes, we choose the maximal violating pair and we also choose the second maximal violating pair in the other one. Thus we make the time spent on training decreased in classification problems. Different from [8], our method is not similar to a single program multiple data model (SPMD) because we choose different working sets in the two parallel processes. Also, our algorithm does not require interaction between the two parallelizing processes because we simultaneously choose the maximal and the second maximal violating pairs and thus do not use the information from each other until they complete their updates.

Then, we provide a description of the parallel algorithm. Recall that problem (3) is to be solved. Based on [6, 7], we are going to solve the object function using its second order information. After initialization, the parallelization phase begins. We denote the maximal violating pair as α_i, α_j and the second maximal violating pair as α_m, α_n. The training algorithm works as follows:

```
Algorithm 1:
begin
  Initialize step:
  Initialize gradient $\nabla Q(\alpha)$ and Lagrange multipliers
  $\alpha_i$, i = 1….l,
  where l is the total number of training instances
  repeat
  Parallel step:
    1: Select working set and choose the maximal violating
    pair $\alpha_i$ and $\alpha_j$
    2: Select working set and choose the second maximal
    violating pair $\alpha_m$ and $\alpha_n$
  Serialization step:
    Update the status of $\alpha_i$ and $\alpha_j$
    Update the status of gradient $\nabla Q(\alpha_i)$ and $\nabla Q(\alpha_j)$
    Update the status of $\alpha_m$ and $\alpha_n$
    Update the status of gradient $\nabla Q(\alpha_m)$ and $\nabla Q(\alpha_n)$
    Compute the object value
  until the optimization criteria are met
Calculate the object value and output the decision func-
tion
end.
```

5 Experiments

The algorithm is implemented on LIBSVM and it shows that the algorithm approximately halves the time cost by the serial LIBSVM algorithm. Furthermore, the loss in accuracy of the parallel algorithm is acceptable. The data sets used are from the web page of LIBSVM [13]. The first data set is the Adult data set, which contains 14 features originally and is preprocessed according to [4]. The number of training examples range from 1605 to 16100 and the number of test examples range from 16461 to 30956. It shows that there is a speedup at almost 50%. We also examine data sets from [3], which show nearly the same result as the Adult data set. We also test our algorithm on small data sets such as splice from Delve and SVMguide from [11], obtaining results similar to the above ones.

Table 1. Experiments on data sets with test examples

Data Sets	Size of training examples	Size of test examples	Accuracy (LIBSVM / Parallel SVM)	Time Elapsed (LIBSVM / Parallel SVM) (ms)
a1a(UCI)	1605	30956	84.4166% / 84.2422%	483 /220
a2a(UCI)	2265	30296	84.592% / 84.5887%	936 /562
a4a(UCI)	4781	27780	84.5392% / 84.5887%	3588 / 1482
a5a(UCI)	6414	26147	84.4227% / 84.4418%	6599 / 3197
a6a(UCI)	11220	21341	84.4806% / 84.4853%	20561/ 11451
a7a(UCI)	16100	16461	84.8065% / 84.7336%	42292/20878
w1a (JP98a)	2477	47272	97.9121% / 97.9311%	297 / 109
w2a (JP98a)	3470	46279	98.0704% / 98.0704%	546 / 298
w3a (JP98a)	4912	44837	98.2314% / 98.1823%	904 / 544
w4a (JP98a)	7366	42383	98.1667% / 98.1667%	1591 / 998
w5a (JP98a)	9888	39861	98.2063% / 98.1987%	2543 / 1219
w6a (JP98a)	17188	32561	98.4184% / 98.4184%	8065 / 4322
splice(Delve)	1000	2175	88.5517% / 88.5517%	483 / 264
svmguide(CWH03a)	3089	4000	93.025% / 94.075%	4805 / 923

There are also some data sets which do not include specific data for testing. Thus, we split the whole data set into 7 pieces randomly and choose 6 of the 7 pieces of data for training and the rest one for testing. Because such preprocessing is done every time before running an experiment, the data is quite reliable. We average the results from the sum of the 10 experiments.

Table 2. Experiments on data sets without test examples

Data sets	Size of training examples	Accuracy (LIBSVM / Parallel SVM)	Time Elapsed (LIBSVM / Parallel SVM) (ms)
breast-cancer(scaled)	683	96.30712% / 96.18366%	9.4 / 9.3
diabetes(scaled)	768	76.38394% / 75.78623%	71.7 / 48.5
heart(scaled)	270	80.71581% / 81.27854%	26.6 / 9.5

From the table above, it shows that the time cost by LIBSVM is almost the same as the time cost by the parallel algorithm in breast-cancer data set from UCI. Actually, the above three data set does not contain a large number of examples, leading to short running time in both LIBSVM and parallel algorithm. The accuracy is also quite acceptable as we can see from above. In diabetes, the LIBSVM outperforms the parallel algorithm by less than 0.6% in prediction while they are very close to each other in the rest two data sets.

We next do our experiments on the MNIST data set which consists of 60000 instances for training examples and 10000 for testing. Actually, training MNIST is a problem of multiclass-classification since the data from MNIST is from 10 classes. According to [7], “one against one” strategy is used in such classification problems, so k(k-1)/2 classifiers are created where k is the number of classes. In MNIST, this means that 45 classifiers will be constructed before the training model is built.

We average the sum of all the running time in the training process and then do the prediction. The average time and the accuracy are listed in the table below and we find that the training time is reduced by 53.45% and the loss in prediction accuracy is about 0.08%. Therefore, the result is quite acceptable.

Table 3. Experiment on multi-class data sets

Data sets	**Size of training examples**	**Size of test examples**	**Accuracy (LIBSVM /Parallel SVM)**	**Time Elapsed (LIBSVM / Parallel SVM) (ms)**
MNIST	60000	10000	98.21% / 98.13%	15687.8 / 7302.178

6 Discussion

Based on the algorithm in [6, 7], we proposed a parallel method of training support vector machines. We choose and update both the maximal violating pair and the second maximal violating pair in the working set, approximately halving the time consumed by LIBSVM and maintaining the prediction accuracy in most cases.

However, in some cases the algorithm fails to converge. Although some of the problems are solved after tuning the parameter C and gamma, the convergence of the algorithm is not confirmed yet. In future works, we will attempt to make the algorithm surely converge by introducing the idea of function gain [12] into our work. After a considerable gain from the parallel step of our algorithm is obtained, we will stop the parallel phase of the algorithm and come back again to run LIBSVM which is serial so as to guarantee the convergence of our algorithm.

Acknowledgment. This work is supported by the National Natural Science Foundation of China via the Grant No. 60903092.

References

1. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144–152. ACM (July 1992)
2. Vapnik, V.: The nature of statistical learning theory. Springer (2000)
3. Platt, J.: Sequential minimal optimization: A fast algorithm for training support vector machines (1998)
4. Osuna, E., Freund, R., Girosi, F.: An improved training algorithm for support vector machines. In: Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing VII, pp. 276–285. IEEE (September 1997)
5. Joachims, T.: Making large-scale support vector machine learning practical. In: Advances in Kernel Methods, pp. 169–184. MIT Press (February 1999)
6. Fan, R.E., Chen, P.H., Lin, C.J.: Working set selection using second order information for training support vector machines. The Journal of Machine Learning Research 6, 1889–1918 (2005)

7. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) 2(3), 27 (2011)
8. Cao, L.J., Keerthi, S.S., Ong, C.J., Zhang, J.Q., Periyathamby, U., Fu, X.J., Lee, H.P., Periyathamby, U., Fu, X.J., Lee, H.P.: Parallel sequential minimal optimization for the training of support vector machines. IEEE Transactions on Neural Networks 17(4), 1039–1049 (2006)
9. Graf, H.P., Cosatto, E., Bottou, L., Dourdanovic, I., Vapnik, V.: Parallel support vector machines: The cascade SVM. In: Advances in Neural Information Processing Systems, pp. 521–528 (2004)
10. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: Improvements to Platt's SMO algorithm for SVM classifier design. Neural Computation 13(3), 637–649 (2001)
11. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification (2003)
12. Glasmachers, T., Igel, C.: Maximum-gain working set selection for SVMs. The Journal of Machine Learning Research 7, 1437–1466 (2006)
13. LIBSVM–A library for Support Vector Machines, `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`